

Unit testing logic


What is it for?

A unit test is used to check whether an isolated part of business logic is (still) correct.

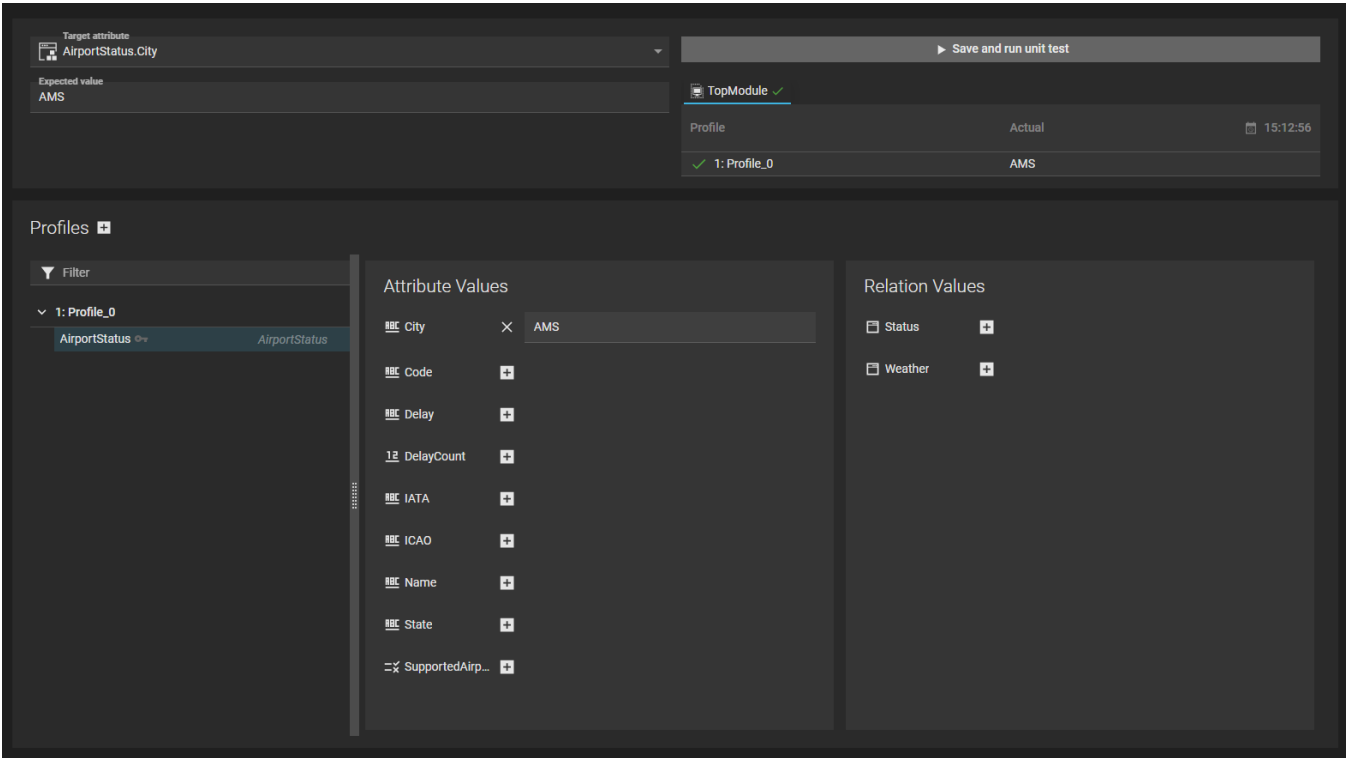
What does it do?

In an [attribute](#) unit test, the actual output value of a target attribute is compared with its expected output value given the input profile and defined business logic ([business rule](#), [decision table](#), [data rule](#), default value) in the module scope.

A unit test is saved and can be repeated, allowing to test business logic if changes are made. Also a unit test can be used as a form of documentation. By defining input and expected output, the functional operation is documented.

To create, delete or open a unit test, the unit test overview should be opened first. This can be done by clicking the unit test icon in the bottom left: .

An example of an attribute unit test:



Property	Description
Target attribute	The attribute of which the actual value is going to be compared with the expected value
Expected value(s)	The expected value of the attribute given the input (profiles). If the target attribute is multivalued, multiple expected values can be entered.
Profiles	The input for the tested element, represented as one or more profiles.
Save and run unit test	Allows running the unit test and shows the result

Profiles

To create a profile, use the + button next to "Profiles". Sometimes different inputs for a unit should lead to the same output. Adding multiple profiles to the unit test makes it possible to assign different profiles to one unit test, all leading to the same output.

After selecting a **profile** on the left in the profiles section:

- The profile can be given a more meaningful name

- The target instance can be selected that determines the starting point of the inference mechanism to determine the value of the attribute by the business logic that is defined for that attribute.
- New instances can be created:
 - From a dropdown
 - All singletons that exist in the module scope can be generated at once

After selecting an **instance** on the left in the profiles section:

- User-set values can be assigned to attributes and relations in the profile
- From the relation value dropdown, new instances can be generated

Running a unit test

Running a unit test means checking if the actual output value(s) of the unit test, given the input profile(s), matches the expected output value(s). There are two ways to run a unit test in Blueriq Encore: from the unit test itself and from the unit test overview.


Run a unit test from the unit test itself

By clicking the Save and run unit test button, the unit test is first saved and then executed.

- If the unit test **succeeds**, meaning that for all profiles the output is as expected, a green checkmark is displayed.
- If the unit test **fails** on one or more profiles, a red exclamation mark is displayed.

When running a unit test from the unit test editor, the test may be run from two different module scopes. Since a unit test is a module element as well, there may be business logic in a higher level module implemented changing the actual output value. Blueriq Encore will automatically detect if this may be the case and will then run the unit test from different relevant module scopes. A tab selector above the results table will then appear, in which the results from different module scopes may be viewed.

Running unit tests from the unit test overview

Opening the unit test overview can be done by clicking the unit test icon in the bottom left: .

When pressing the button "Run all unit tests in module", all unit tests that are in the module scope displayed on the right will be run at once.

When pressing the play button next to an attribute name, only unit tests for a specific attribute will run,

After running unit tests from the overview, a green checkmark or red exclamation mark will be shown. Selecting the unit test in the list on the left will show the expected and actual output value for each profile.

Unit test coverage

The Blueriq Model Analyzer (BMA) has the ability to calculate a unit test coverage for all decision tables and business rules within a project/module. [Read more about it here.](#)